Journal of Mathematical Research with Applications
Jan., 2017, Vol. 37, No. 1, pp. 97–106
DOI:10.3770/j.issn:2095-2651.2017.01.009
Http://jmre.dlut.edu.cn

A Cost-Efficient Variant of the Incremental Newton Iteration for the Matrix pth Root

Fuminori TATSUOKA*, **Tomohiro SOGABE, Yuto MIYATAKE, Shaoliang ZHANG**

Department of Computational Science and Engineering, Graduate School of Engineering, Nagoya University, Nagoya 464-8603*, Japan*

Dedicated to Professor Renhong WANG on the Occasion of His Eightieth Birthday

Abstract Incremental Newton (IN) iteration, proposed by Iannazzo, is stable for computing the matrix pth root, and its computational cost is $\mathcal{O}(n^3 p)$ flops per iteration. In this paper, a cost-efficient variant of IN iteration is presented. The computational cost of the variant well agrees with $\mathcal{O}(n^3 \log p)$ flops per iteration, if p is up to at least 100.

Keywords matrix pth root; matrix polynomial

MR(2010) Subject Classification 65F30; 65F60; 65H04

1. Introduction

A matrix pth root ($p \in \mathbb{N}$) of $A \in \mathbb{C}^{n\times n}$ is defined as a solution of the following matrix equation:

$$X^p = A.$$

While this matrix equation might have infinitely many solutions, the target of this paper is a solution whose eigenvalues lie in the set $\{z \in \mathbb{C} \setminus \{0\} : -\pi/p < \arg z < \pi/p\}$. If A has no nonpositive real eigenvalues, the target solution is unique [1, Theorem 7.2] and is referred to as the principal matrix pth root of A, denoted by the symbol $A^{1/p}$. Throughout this paper, A is assumed to have no nonpositive real eigenvalues. The principal matrix pth root arises in lattice quantum chromodynamics (QCD) calculations [2] and in the computation of the matrix logarithm [1] that corresponds to the inverse function of the matrix exponential. Therefore, numerical algorithms for computing the principal matrix pth root have been developed during the past decade.

Numerical algorithms for the principal matrix pth root can be classified roughly into direct methods and iterative methods. Direct methods include, for example, the Schur method [3], the matrix sign method [4], and a method based on repeated eigenvalues of A (see [5]). The Schur

Received October 31, 2016; Accepted December 7, 2016
Supported by JSPS KAKENHI (Grant No. 26286088).
* Corresponding author
E-mail address: f-tatsuoka@na.nuap.nagoya-u.ac.jp (Fuminori TATSUOKA); sogabe@na.nuap.nagoya-u.ac.jp (Tomohiro SOGABE); miyatake@na.nuap.nagoya-u.ac.jp (Yuto MIYATAKE); zhang@na.nuap.nagoya-u.ac.jp (Shaoliang ZHANG)

method can be performed in $\mathcal{O}(n^3 p)$ flops, the matrix sign method can be performed in at least $\mathcal{O}(n^3 p \log p)$ flops, and the computational cost of the method based on repeated eigenvalues is not explicitly stated in [5]. Therefore, in terms of computational cost, the Schur method is likely the method of choice for large-scale problems. Iterative methods include Newton's method and Halley's method for $A^{1/p}$, proposed by Iannazzo [6,7], and Newton's method for $A^{-1/p}$, proposed by Guo [8]. In this paper, we consider Newton's method for $A^{1/p}$, since that method is the most fundamental iterative method. In addition, it has been reported that Newton's method for $A^{1/p}$ gives a more accurate solution than the Schur method for some ill-conditioned matrices [6].

Now, let us recall several results for Newton's method by Iannazzo [6]. It is known that Newton's method for a matrix pth root can be written as

$$X_{k+1} = \frac{(p-1)X_k + AX_k^{1-p}}{p}, \quad k = 0, 1, 2, \ldots, \tag{1.1}$$

with an initial guess X_0 satisfying $AX_0 = X_0 A$. However, it is not always guaranteed that this method converges to the principal pth root. Iannazzo showed that if both of the following conditions,

$$\begin{cases} \text{all eigenvalues of } A \text{ lie in the set } \{z \in \mathbb{C} : \mathrm{Re}\ z > 0, |z| \leq 1\}, & (1.2) \\ X_0 = I, & (1.3) \end{cases}$$

are satisfied, then Newton's method (1.1) converges to $A^{1/p}$. Next, Iannazzo proposed a preconditioning step, computing $\tilde{A} = A^{1/2}/\|A^{1/2}\|$ with a consistent norm (say, p-norm, Frobenius norm), because then $\tilde{A}$ satisfies the condition (1.2) for any A. Even if the matrix A is preconditioned, Newton's iteration (1.1) could be unstable in the neighborhood of $A^{1/p}$ (see [3]). Then, Iannazzo proposed three stable iterations:

$$\begin{cases} X_{k+1} = X_k + H_k, \ F_k = X_k X_{k+1}^{-1}, \\ H_{k+1} = -\dfrac{1}{p} H_k \Big(\displaystyle\sum_{i=0}^{p-2} (i+1) X_{k+1}^{-1} F_k^i \Big) H_k, \end{cases} \quad \Big(X_0 = I, \ H_0 = \frac{A-I}{p}\Big) \tag{1.4}$$

$$\begin{cases} X_{k+1} = X_k + H_k, \ F_k = X_k X_{k+1}^{-1}, \\ H_{k+1} = -X_k \Big(\dfrac{I - F_k^p}{p} + F_k^{p-1}(F_k - I) \Big), \end{cases} \quad \Big(X_0 = I, \ H_0 = \frac{A-I}{p}\Big) \tag{1.5}$$

and

$$\begin{cases} X_{k+1} = X_k \Big(\dfrac{(p-1)I + N_k}{p} \Big), \\ N_{k+1} = \Big(\dfrac{(p-1)I + N_k}{p} \Big)^{-p} N_k. \end{cases} \quad (X_0 = I, \ N_0 = A) \tag{1.6}$$

In particular, iteration (1.4) is called incremental Newton (IN) iteration, and iteration (1.6) is called coupled Newton iteration.

It is known that Newton's method converges quadratically in a neighborhood of the solution, but global convergence of that method is not guaranteed. One way to globalize the convergence

of Newton's method is by using damping.[1] From this point of view, it might be possible to apply damping to IN iteration (1.4) and iteration (1.5). Comparing these two iterations, the cost of IN iteration (1.4) is $\mathcal{O}(n^3 p)$ flops per iteration, higher than $\mathcal{O}(n^3 \log p)$ flops for iteration (1.5). On the other hand, the incremental part of IN iteration (1.4) is computed in the form of $H_{k+1} = f_k(H_k)$, in contrast to iteration (1.5). This characteristic of IN iteration (1.4) might provide a new viewpoint for convergence analysis to confirm that H_k converges to O. That is to say, if H_{k+1} explicitly includes H_k, then H_{k+1} is represented as $H_{k+1} = (f_k \circ f_{k-1} \circ \cdots \circ f_0)(H_0)$, and its convergence behavior might be analyzed using composite mapping $(f_k \circ f_{k-1} \circ \cdots \circ f_0)$ and initial matrix H_0. Thus, IN iteration (1.4) is worth considering.

The purpose of this paper is to provide a cost-efficient variant of IN iteration (1.4) whose increment part is computed in the form $H_{k+1} = f_k(H_k)$. In this paper, we reduce the cost of IN iteration (1.4) by finding a specific matrix polynomial in IN iteration (1.4) and proposing a decomposition of the matrix polynomial.

The remainder of this paper is organized as follows. In Section 2, a variant of IN iteration is shown, and we numerically estimate its cost at $\mathcal{O}(n^3 \log p)$ flops per iteration. In Section 3, we present the results of numerical experiments. We conclude in Section 4.

2. Variant of IN iteration

The computational cost for computing the increment part

$$H_{k+1} = -\frac{1}{p} H_k \Big(\sum_{i=0}^{p-2} (i+1) X_{k+1}^{-1} F_k^i \Big) H_k \tag{2.1}$$

is the highest in IN iteration (1.4), because $(2p + 2/3)n^3 + \mathcal{O}(n^2)$ flops are required for Eq. (2.1), and $(2p + 10/3)n^3 + \mathcal{O}(n^2)$ flops for IN iteration (1.4). In this section, without losing the previous matrix H_k, Eq. (2.1) is rewritten to reduce the number of matrix multiplications whose computational costs are $\mathcal{O}(n^3)$ flops.

2.1. Rewriting the increment

From the definition of IN iteration (1.4), the increment H_k is equivalent to $X_{k+1} - X_k$, and thus

$$H_k X_{k+1}^{-1} = (X_{k+1} - X_k) X_{k+1}^{-1} = I - F_k.$$

Substituting this relation into Eq. (2.1) yields

$$\begin{aligned} H_{k+1} &= -\frac{1}{p} H_k X_{k+1}^{-1} \Big(\sum_{i=0}^{p-2} (i+1) F_k^i \Big) H_k = -\frac{1}{p} (I - F_k) \Big(\sum_{i=0}^{p-2} (i+1) F_k^i \Big) H_k \\ &= -\frac{1}{p} [I + F_k + F_k^2 + \cdots + F_k^{p-2} - (p-1) F_k^{p-1}] H_k \\ &= -\frac{1}{p} \big\{ [-(p-1)F_k + pI][I + F_k + F_k^2 + \cdots + F_k^{p-2}] - (p-1) I \big\} H_k. \end{aligned}$$

[1] A damped Newton iteration is represented as $X_{k+1} = X_k + \alpha_k H_k$ ($\alpha_k \in (0, 1]$), where α_k is a relaxation factor chosen to reduce residuals.

Introducing the matrix polynomial $P_d(X) := I + X + X^2 + \cdots + X^d$, enables Eq. (2.1) to be simplified further to

$$H_{k+1} = -\frac{1}{p}\{[-(p-1)F_k + pI]P_{p-2}(F_k) - (p-1)I\}H_k. \tag{2.2}$$

The number of matrix multiplications for Eq. (2.2) is equal to the number of matrix multiplications for $P_{p-2}(F_k)$ plus two. We now define a variant of IN iteration as

$$\begin{cases} X_{k+1} = X_k + H_k, \ F_k = X_k X_{k+1}^{-1}, \\ H_{k+1} = -\dfrac{1}{p}\{[-(p-1)F_k + pI]P_{p-2}(F_k) - (p-1)I\}H_k. \end{cases} \tag{2.3}$$

This new expression motivates us to reduce the number of matrix multiplications for computing $P_{p-2}(F_k)$.

Furthermore, this variant (2.3) is as stable as original IN iteration (1.4). We use the following definition of stability to analyze the variant (2.3).

Definition 2.1 ([1, Definition 4.17]) *Consider an iteration $X_{k+1} = g(X_k)$ with a fixed point X. Assume that g is Fréchet differentiable at X. The iteration is stable in a neighborhood of X if the Fréchet derivative $L_g(X)$ has bounded powers, that is, there exists a constant c such that $\|L_g^i(X)\| \leq c$ for all $i > 0$.*

In Definition 2.1, $L_g^i(X)$ is ith power of the Fréchet derivative L at X. For more details of definitions of $L_g^i(X)$, $\|L_g^i(X)\|$, and other notations used for stability analysis, see Appendix. Then, we show that the variant (2.3) is stable.

Proposition 2.2 *The variant (2.3) is stable.*

Proof The iteration function for the variant (2.3) is

$$G\left(\begin{bmatrix} X \\ H \end{bmatrix}\right) = \begin{bmatrix} X + H \\ -\frac{1}{p}\{[-(p-1)F + pI]P_{p-2}(F) - (p-1)I\}H \end{bmatrix} \quad (F = X(X+H)^{-1}), \tag{2.4}$$

and the fixed point is $\left[\begin{smallmatrix} A^{1/p} \\ O \end{smallmatrix}\right]$. In order to calculate the Fréchet derivative of G at $\left[\begin{smallmatrix} A^{1/p} \\ O \end{smallmatrix}\right]$, we calculate $G\left(\left[\begin{smallmatrix} A^{1/p} \\ O \end{smallmatrix}\right]\right)$ and $G\left(\left[\begin{smallmatrix} A^{1/p}+E_X \\ O+E_H \end{smallmatrix}\right]\right)$, where $\|E_X\|$ and $\|E_H\|$ are sufficiently small. Substituting $X = A^{1/p}$ and $H = O$ into Eq. (2.4),

$$G\left(\begin{bmatrix} A^{1/p} \\ O \end{bmatrix}\right) = \begin{bmatrix} A^{1/p} \\ -\frac{1}{p}\{[-(p-1)I + pI][\sum_{n=0}^{p-2} I] - (p-1)I\}O \end{bmatrix} = \begin{bmatrix} A^{1/p} \\ O \end{bmatrix}, \tag{2.5}$$

and substituting $X = A^{1/p} + E_X$ and $H = O + E_H$ into Eq. (2.4),

$$\begin{aligned} G\left(\begin{bmatrix} A^{1/p} + E_X \\ O + E_H \end{bmatrix}\right) &= \begin{bmatrix} A^{1/p} + E_X + E_H \\ -\frac{1}{p}\{[-(p-1)F_\Delta + pI][\sum_{i=0}^{p-2} F_\Delta^i] - (p-1)I\}E_H \end{bmatrix} \\ &\qquad (F_\Delta = (A^{1/p} + E_X)(A^{1/p} + E_X + E_H)^{-1}) \\ &= \begin{bmatrix} A^{1/p} + E_X + E_H \\ -\frac{1}{p}[I + F_\Delta + F_\Delta^2 + \cdots + F_\Delta^{p-2} - (p-1)F_\Delta^{p-1}]E_H \end{bmatrix}. \end{aligned} \tag{2.6}$$

Since $\|E_X\|$ and $\|E_H\|$ are sufficiently small, F_Δ becomes

$$F_\Delta = (A^{1/p} + E_X)(A^{1/p} + E_X + E_H)^{-1}$$

$$= [A^{1/p} + E_X][A^{-1/p} - A^{-1/p}(E_X + E_H)A^{-1/p} + \mathcal{O}(\|E_X + E_H\|^2)]$$
$$= I - E_H A^{-1/p} + \mathcal{O}(\|E_X\|^2) + \mathcal{O}(\|E_H\|^2) + \mathcal{O}(\|E_X\|\|E_H\|). \tag{2.7}$$

Using Eq. (2.7), F_Δ^i becomes

$$F_\Delta^i = (I - E_H A^{-1/p} + \mathcal{O}(\|E_X\|^2) + \mathcal{O}(\|E_H\|^2) + \mathcal{O}(\|E_X\|\|E_H\|))^i$$
$$= I - iE_H A^{-1/p} + \mathcal{O}(\|E_X\|^2) + \mathcal{O}(\|E_H\|^2) + \mathcal{O}(\|E_X\|\|E_H\|).$$

Therefore, the lower part of (2.6) can be rewritten as

$$\begin{aligned}
&-\frac{1}{p}[I + F_\Delta + F_\Delta^2 + \cdots + F_\Delta^{p-2} - (p-1)F_\Delta^{p-1}]E_H \\
&= -\frac{1}{p}\big[I + (I - E_H A^{-1/p}) + (I - 2E_H A^{-1/p}) + \cdots + (I - (p-2)E_H A^{-1/p}) - \\
&\quad (p-1)(I - (p-1)E_H A^{-1/p}) + \mathcal{O}(\|E_X\|^2) + \mathcal{O}(\|E_H\|^2) + \mathcal{O}(\|E_X\|\|E_H\|)\big]E_H \\
&= -\frac{1}{p}[\frac{p(p-1)}{2}E_H A^{-1/p} + \mathcal{O}(\|E_X\|^2) + \mathcal{O}(\|E_H\|^2) + \mathcal{O}(\|E_X\|\|E_H\|)]E_H \\
&= \mathcal{O}(\|E_X\|^2) + \mathcal{O}(\|E_H\|^2),
\end{aligned}$$

and we have

$$G\left(\begin{bmatrix} A^{1/p} + E_X \\ O + E_H \end{bmatrix}\right) = \begin{bmatrix} A^{1/p} + E_X + E_H \\ \mathcal{O}(\|E_X\|^2) + \mathcal{O}(\|E_H\|^2) \end{bmatrix}. \tag{2.8}$$

From Eqs. (2.5) and (2.8), it holds that

$$G\left(\begin{bmatrix} A^{1/p} + E_X \\ O + E_H \end{bmatrix}\right) - G\left(\begin{bmatrix} A^{1/p} \\ O \end{bmatrix}\right) - \begin{bmatrix} I & I \\ O & O \end{bmatrix}\begin{bmatrix} E_X \\ E_H \end{bmatrix}$$
$$= \begin{bmatrix} O \\ \mathcal{O}(\|E_X\|^2) + \mathcal{O}(\|E_H\|^2) \end{bmatrix} = o\left(\left\|\begin{bmatrix} E_X \\ E_H \end{bmatrix}\right\|\right),$$

and we obtain

$$L_G\left(\begin{bmatrix} A^{1/p} \\ O \end{bmatrix}, \begin{bmatrix} E_X \\ E_H \end{bmatrix}\right) = \begin{bmatrix} I & I \\ O & O \end{bmatrix}\begin{bmatrix} E_X \\ E_H \end{bmatrix}.$$

The matrix $\left[\begin{smallmatrix} I & I \\ O & O \end{smallmatrix}\right]$ is idempotent because

$$\begin{bmatrix} I & I \\ O & O \end{bmatrix}^2 = \begin{bmatrix} I & I \\ O & O \end{bmatrix}.$$

Then, for all $i > 0$, $\left\|L_G^i\left(\left[\begin{smallmatrix} A^{1/p} \\ O \end{smallmatrix}\right]\right)\right\|$ is bounded. From the above, the variant (2.3) is stable.[2] □

In the next subsection, we provide a means of reducing matrix multiplications of $P_{p-2}(F_k)$.

2.2. Decomposition of the polynomial

If $d \geq 3$, the matrix polynomial $P_d(X)$ can be rewritten in a more efficient form:

$$P_d(X) = \begin{cases} P_{\frac{d-1}{2}}(X^2) \cdot (X + I) & (d \text{ is odd}) \\ P_{\frac{d-2}{2}}(X^2) \cdot (X^2 + X) + I & (d \text{ is even}). \end{cases} \tag{2.9}$$

On the right-hand side of Eq. (2.9), there is a new matrix polynomial whose variable is X^2 and degree is approximately half of d. This decomposition reduces the number of matrix

[2] The stability of IN iteration (1.4) can be proved in a similar manner.

multiplications by almost a factor of two. Thus, the number of matrix multiplications of $P_d(X)$ is reduced by applying the decomposition (2.9) to $P_d(X)$, repeatedly.

Let us show the example of $d = 57$.[3]

$$\begin{aligned} P_{57}(X) &= I + X + X^2 + \cdots + X^{57} \qquad (2.10)\\ &= \{P_{28}(X^2)\}\{X + I\}\\ &= \{P_{13}(X^4)(X^4 + X^2) + I\}\{X + I\}\\ &= \{P_6(X^8)(X^4 + I)(X^4 + X^2) + I\}\{X + I\}\\ &\quad\vdots\\ &= \{[(X^{32} + X^{16} + I)(X^{16} + X^8) + I][X^4 + I][X^4 + X^2] + I\}\{X + I\}. \qquad (2.11) \end{aligned}$$

In this example, $P_{57}(X)$ of Eq. (2.10) is computed using 56 matrix multiplications by naive implementation. On the other hand, after applying the decomposition (2.9) to Eq. (2.10) four times, Eq. (2.11) can be computed with nine matrix multiplications. In detail, five matrix multiplications are required for constructing five intermediate matrices, X^2, X^4, X^8, X^{16}, and X^{32}, and another four matrix multiplications are required for multiplication of the subpolynomials.

Finally, we combine variant (2.3) with decomposition (2.9) into Algorithm 1 for practice.

Algorithm 1 Newton's method with the variant of IN iteration

Input: $A \in \mathbb{C}^{n\times n}$ (Satisfying condition (1.2) in section 1), $p \in \mathbb{N}$
Output: $X \approx A^{1/p}$

1: Decompose P_{p-2} by applying the decomposition (2.9), repeatedly.
2: $X_0 \leftarrow I$ ($\because$ Condition (1.3)), $H_0 \leftarrow \frac{A-I}{p}$
3: **for** $k = 0, 1, 2, \ldots$ until convergence **do**
4: $\quad X_{k+1} = X_k + H_k$
5: $\quad F_k = X_k X_{k+1}^{-1}$
6: $\quad$ Compute $P_{p-2}(F_k)$
7: $\quad H_{k+1} = -\frac{1}{p}\{[-(p-1)F_k + pI]P_{p-2}(F_k) - (p-1)I\}H_k$
8: $X \leftarrow X_k$

2.3. Estimation of the computational cost of the variant

We calculated the computational cost of the variant (2.3) for $p \in [5, 100]$ numerically and found that cost to be consistent with $(2\lfloor 2\log_2(p-1)\rfloor + 8/3)n^3$. Here, the computational cost $8/3n^3$ results from the computation of $F_k (= X_k X_{k+1}^{-1})$ by using the LU decomposition of X_{k+1}. While a proof that the cost of variant (2.3) is $\mathcal{O}(n^3 \log p)$ flops per iteration is left for future work, this numerical result agrees with that expectation. In addition, we calculated the costs of IN iteration (1.4) and the iteration (1.5) for $p \in [5, 100]$ to compare them with that of variant (2.3). The result is shown in Figure 1.

[3] The polynomial $P_{57}(F_k)$ appears when calculating the matrix 59th root.

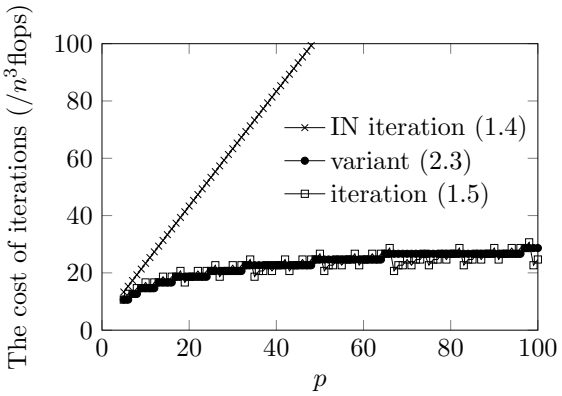


Figure 1 The computational costs per iteration for the three iterations

It is clear from the figure that the computational cost of the variant (2.3) is lower than that of IN iteration (1.4) and competitive with that of iteration (1.5). For example, when $d = 59$, the computational cost of variant (2.3) is approximately a quarter of that of IN iteration (1.4) and slightly higher than that of iteration (1.5).

3. Numerical experiment

This section describes a numerical experiment in which the principal 59th roots of test matrices are calculated. The test matrices are described in Table 1.

Test matrix A (Matrix ID)	Size	Non-zero elements	cond(A)	Symmetry Property
msc01440 [9] (1)	1440	44998	3.3×10^6	Symmetric positive define
Random matrix (2)	1500	2250000	3.8×10^2	Symmetric positive define
NNC1374 [10] (3)	1374	8606	3.7×10^{14}	Unsymmetric

Table 1 Test matrices

First, we preconditioned the test matrices to satisfy the sufficient condition (1.2) of global convergence in Section 1: all eigenvalues of A lie in the set $\{z \in \mathbb{C} : \mathrm{Re}\ z > 0, |z| \leq 1\}$. Thus, we computed $\tilde{A} = A^{1/2}/\|A^{1/2}\|_F$. Then, we computed $\tilde{A}^{1/59}$ by IN iteration (1.4), variant (2.3) of Algorithm 1, and iteration (1.5). The computational costs of these three iterations are shown in Table 2.

Iteration	Computational costs per iteration(flops)
IN iteration (1.4)	$(118 + 10/3)n^3 + \mathcal{O}(n^2)$
variant (2.3)	$(22 + 8/3)n^3 + \mathcal{O}(n^2)$
iteration (1.5)	$(20 + 8/3)n^3 + \mathcal{O}(n^2)$

Table 2 Computational costs for computing the principal 59th root

For this experiment, Python 3.5 was used for programming, and Intel(R) Core™ i7 2.8GHz CPU and 8GB RAM were used for computation.

First, Figure 2 shows the ratios of computation time of these three iterations. From Figure 2, the computation time of variant (2.3) is approximately one fourth of that of IN iteration (1.4) and slightly longer than that of (1.5) in all cases. Here it can be seen that both the computation time and the computational cost decreased.

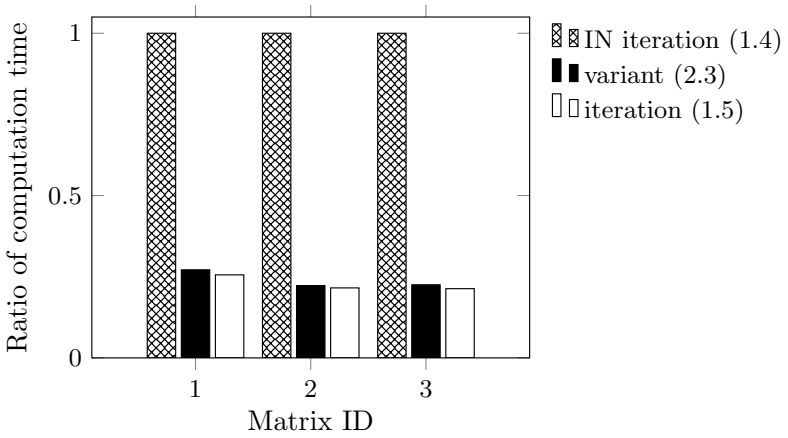


Figure 2 Time comparison of the three iterations

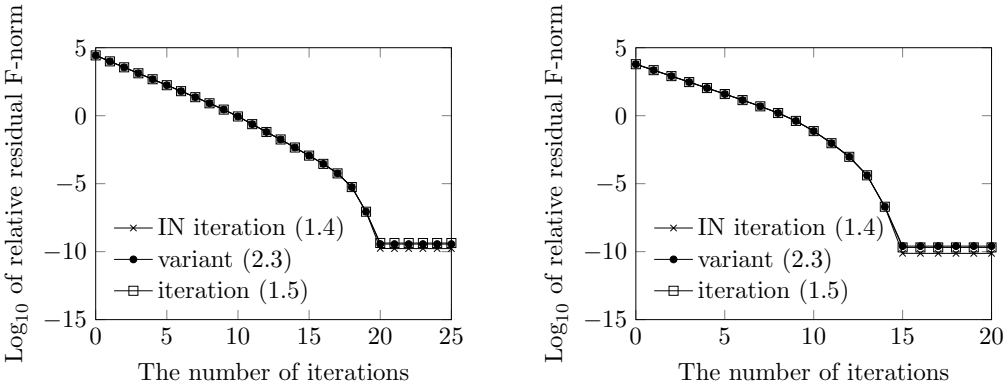
Next, Figure 3 shows the relative residual defined as $R(X) = \|X^p - A\|_F / \|A\|_F$ for these three iterations. The figure shows that the convergence behavior of variant (2.3) differs little from that of IN iteration (1.4) and iteration (1.5). Since there is some possibility of numerical cancellation of variant (2.3), IN iteration (1.4) is slightly better than variant (2.3) in terms of accuracy.

4. Conclusion and future work

In this paper, a variant of IN iteration is proposed whose computational cost well agreed with $\mathcal{O}(n^3 \log p)$ flops per iteration if p is up to at least 100, and whose increment part still has the form $H_{k+1} = f_k(H_k)$. We have learned from the results of the numerical experiment that the variant is competitive with iteration (1.5) in terms of accuracy and computation time. The proposed variant therefore becomes a choice for practical application.

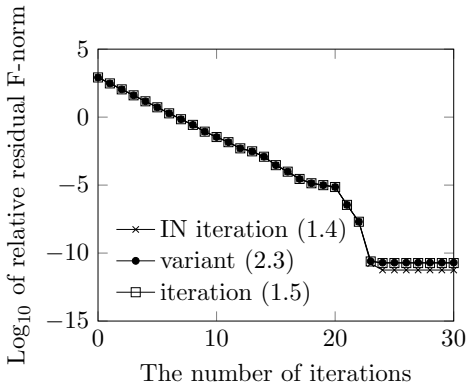
The most important future work is to prove that the computational cost of the variant is $\mathcal{O}(n^3 \log p)$. Other future work includes reducing the computation time of Newton's method for the principal matrix pth root by reducing the number of iterations. However, it is not clear how to choose a better initial guess than the conventional initial guess I (the identity matrix). It might be easier to find a good initial guess, when considering the damped Newton method.

Acknowledgements The authors are grateful to the reviewer for the careful reading and the comments that substantially enhanced the quality of the manuscript.



(a) Residual Plot of msc01440 (1)

(b) Residual plot of the random matrix (2)



(c) Residual plot of NNC1374 (3)

Figure 3 Residual comparison of the three iterations

Appendix

In this section, we recall some definitions and notations which were given in [1], where we consider the matrix norm is consistent.

1. The notation $X = \mathcal{O}(\|E\|)$ denotes that $\|X\| \leq c\|E\|$ for some constant c for all sufficiently small $\|E\|$, while $X = o(\|E\|)$ means that $\|X\|/\|E\| \to 0$ as $E \to O$ (see [1, p. 321]).

2. The Fréchet derivative of a matrix function $f : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$ at a point $X \in \mathbb{C}^{n\times n}$ is a linear mapping

$$\begin{aligned} \mathbb{C}^{n\times n} &\xrightarrow{L} \mathbb{C}^{n\times n} \\ E &\mapsto L(X, E) \end{aligned}$$

such that for all $E \in \mathbb{C}^{n\times n}$

$$f(X + E) - f(X) - L(X, E) = o(\|E\|).$$

If we need to show the dependence on f we will write $L_f(X, E)$. When we want to refer to the mapping at X and not its value in a particular direction we will write $L(X)$ (see [1, p. 56]).

3. The norm of $L(X)$ is defined by$\|L(X)\| := \max_{Z \neq O} \frac{\|L(X,Z)\|}{\|Z\|}$ (see [1, p. 56]).

4. We write $L^i(X)$ to denote the ith power of the Fréchet derivative L at X, defined as i-fold composition; thus $L^3(X, E) \equiv L\big(X, L(X, L(X, E))\big)$ (see [1, p. 97]).

References

[1] N. J. HIGHAM. *Functions of Matrices: Theory and Computation*. SIAM, Philadelphia, 2008.
[2] M. A. CLARK, A. D. KENNEDY. *Accelerating dynamical-fermion computations using the rational hybrid Monte Carlo algorithm with multiple pseudofermion fields*. Phys. Rev. Lett., 2007, **98**(1): (051601) 1–4.
[3] M. I. SMITH. *A Schur algorithm for computing matrix pth roots*. SIAM J. Matrix Anal. Appl., 2003, **24**(4): 971–989.
[4] D. A. BINI, N. J. HIGHAM, B. MEINI. *Algorithms for the matrix pth root*. Numer. Algorithms, 2005, **39**(4): 349–378.
[5] A. SADEGHI, A. I. M. ISMAIL, A. AHMAD. *Computing the pth roots of a matrix with repeated eigenvalues*. Appl. Math. Sci., 2011, **5**(53): 2645–2661.
[6] B. IANNAZZO. *On the Newton method for the matrix pth root*. SIAM J. Matrix Anal. Appl., 2006, **28**(2): 503–523.
[7] B. IANNAZZO. *A family of rational iterations and its application to the computation of the matrix pth root*. SIAM J. Matrix Anal. Appl., 2008, **30**(4): 1445–1462.
[8] Chunhua GUO, N. J. HIGHAM. *A Schur-Newton method for the matrix pth root and its inverse*. SIAM J. Matrix Anal. Appl., 2006, **28**(3): 788–804.
[9] T. A. DAVIS, Yifan HU. *The University of Florida sparse matrix collection*. ACM Trans. Math. Software, 2011, **38**(1): 1–25.
[10] R. F. BOISVERT, R. POZO, K. REMINGTON, et al. *Matrix Market: A Web Resource for Test Matrix Collections*. Springer US, Boston, MA, 1997.